QR Factorization of Tall and Skinny Matrices in a Grid Computing Environment

Emmanuel AGULLO (INRIA / LaBRI)
Camille COTI (Iowa State University)
Jack DONGARRA (University of Tennessee)
Thomas HÉRAULT (U. Paris Sud / U. of Tennessee / LRI / INRIA)
Julien LANGOU (University of Colorado Denver)

IPDPS, Atlanta, USA, April 19-23, 2010

Question

Can we speed up dense linear algebra applications using a computational grid ?

Building blocks

Tremendous computational power of grid infrastructures

- BOINC: 2.4 Pflop/s,
- Folding@home: 7.9 Pflop/s.

MPI-based linear algebra libraries

- ScaLAPACK;
- HP Linpack.

Grid-enabled MPI middleware

- MPICH-G2;
- PACX-MPI;
- GridMPI.

Past answers

Can we speed up dense linear algebra applications using a computational grid ?

- ⋆ GrADS project [Petitet et al., 2001]:
  - ☺ Grid enables to process larger matrices;
  - ☹ For matrices that can fit in the (distributed) memory of a cluster, the use of a single cluster is optimal.
- ⋆ Study on a cloud infrastructure [Napper et al., 2009] Linpack on Amazon EC2 commercial offer:
  - ☹ Under-calibrated components;
  - ☹ Grid costs too much

Our approach

Principle

Confine intensive communications (ScaLAPACK calls) within the different geographical sites.

Method

Articulate:

- ⋆ Communication-Avoiding algorithms [Demmel et al., 2008];
- ⋆ with a topology-aware middleware (QCG-OMPI).

Focus

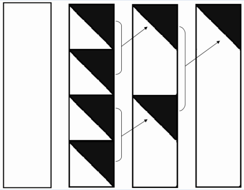
- ⋆ QR factorization;
- ⋆ Tall and Skinny matrices.

Outline

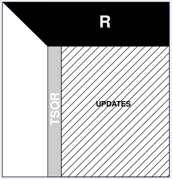
Outline

Communication-Avoiding QR (CAQR) [Demmel et al., 2008]

Tall and Skinny QR (TSQR)



CAQR

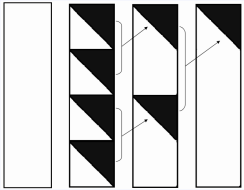


Examples of applications for TSQR

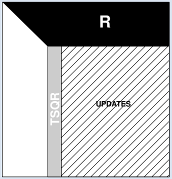
- panel factorization in CAQR;
- block iterative methods (iterative methods with multiple right-hand sides or iterative eigenvalue solvers);
- linear least squares problems with a number of equations extremely larger than the number of unknowns.

Communication-Avoiding QR (CAQR) [Demmel et al., 2008]

Tall and Skinny QR (TSQR)



CAQR

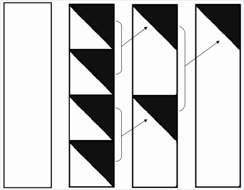


Examples of applications for TSQR

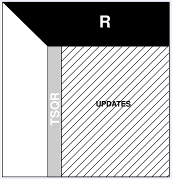
- panel factorization in CAQR;
- block iterative methods (iterative methods with multiple right-hand sides or iterative eigenvalue solvers);
- linear least squares problems with a number of equations extremely larger than the number of unknowns.

Communication-Avoiding QR (CAQR) [Demmel et al., 2008]

Tall and Skinny QR (TSQR)



CAQR

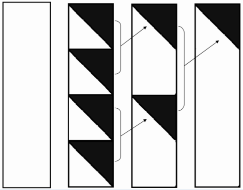


Examples of applications for TSQR

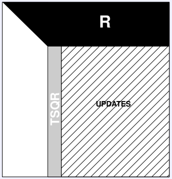
- ⋆ panel factorization in CAQR;
- ⋆ block iterative methods (iterative methods with multiple right-hand sides or iterative eigenvalue solvers);
- ⋆ linear least squares problems with a number of equations extremely larger than the number of unknowns.

Communication-Avoiding QR (CAQR) [Demmel et al., 2008]

Tall and Skinny QR (TSQR)



CAQR



Examples of applications for TSQR

- panel factorization in CAQR;
- block iterative methods (iterative methods with multiple right-hand sides or iterative eigenvalue solvers);
- linear least squares problems with a number of equations extremely larger than the number of unknowns.

Topology-aware MPI middleware for the Grid

MPICH-G2

- description of the topology through the concept of colors:
  - ☺ used to build topology-aware MPI communicators;
  - ☹ the application has to adapt itself to the discovered topology;
- based on MPICH.

QCG-OMPI

- resource-aware grid meta-scheduler (QosCosGrid);
- allocation of resources that match requirements expressed in a "JobProfile" (amount of memory, CPU speed, network properties between groups of processes, . . .)
  - ☺ application always executed on an appropriate resource topology.
- based on OpenMPI.

Topology-aware MPI middleware for the Grid

MPICH-G2

- description of the topology through the concept of colors:
  - ☺ used to build topology-aware MPI communicators;
  - ☹ the application has to adapt itself to the discovered topology;
- based on MPICH.

QCG-OMPI

- resource-aware grid meta-scheduler (QosCosGrid);
- allocation of resources that match requirements expressed in a "JobProfile" (amount of memory, CPU speed, network properties between groups of processes, . . .)
  - ☺ application always executed on an appropriate resource topology.
- based on OpenMPI.

Topology-aware MPI middleware for the Grid

MPICH-G2

- description of the topology through the concept of colors:
  - ☺ used to build topology-aware MPI communicators;
  - ☹ the application has to adapt itself to the discovered topology;
- based on MPICH.

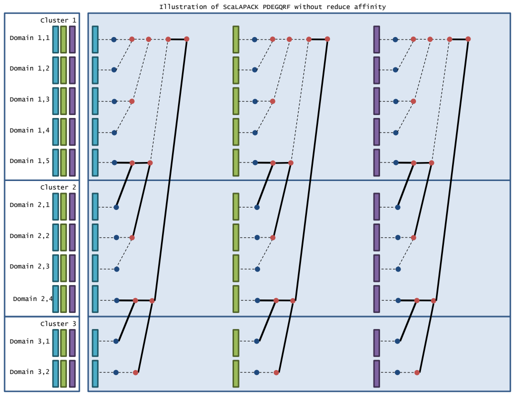
QCG-OMPI

- resource-aware grid meta-scheduler (QosCosGrid);
- allocation of resources that match requirements expressed in a “JobProfile” (amount of memory, CPU speed, network properties between groups of processes, . . .)
  - ☺ application always executed on an appropriate resource topology.
- based on OpenMPI.

Outline

Communication pattern (M-by-3 matrix)

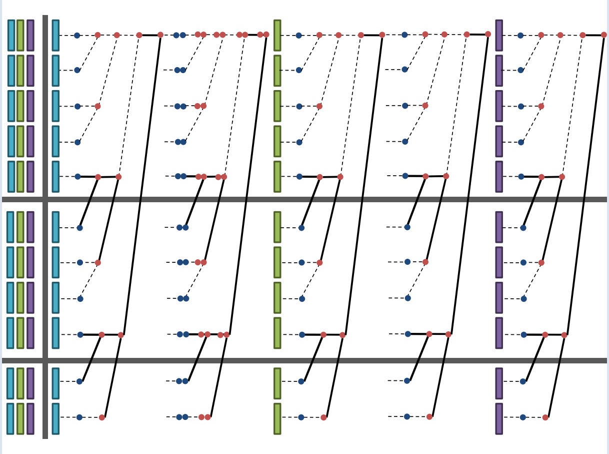
ScaLAPACK (panel factorization routine) - non optimized tree



25 inter-cluster communications

Communication pattern (M-by-3 matrix)

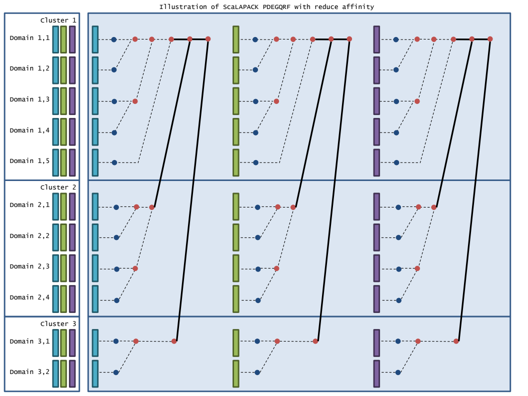
ScaLAPACK (panel factorization routine) - non optimized tree



25 inter-cluster communications

Communication pattern (M-by-3 matrix)

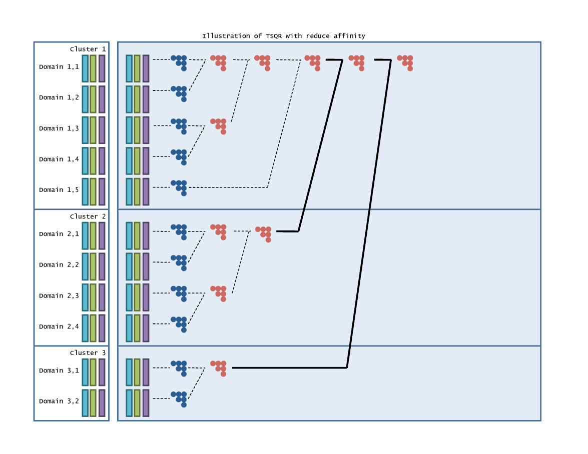
ScaLAPACK (panel factorization routine) - optimized tree



10 inter-cluster communications

Communication pattern (M-by-3 matrix)

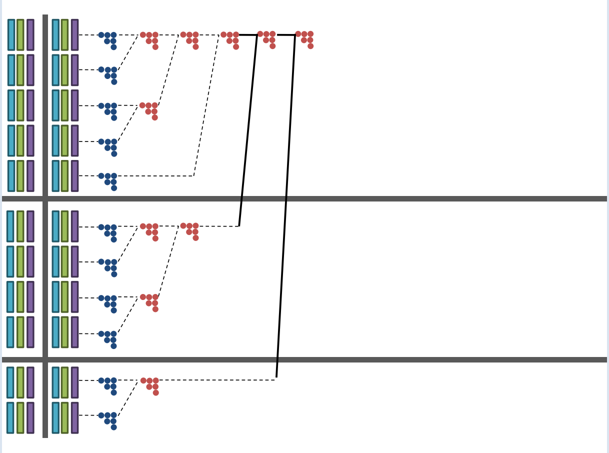
TSQR - optimized tree



2 inter-cluster communications

Communication pattern (M-by-3 matrix)

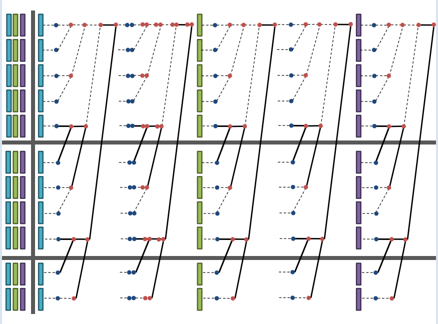
TSQR - optimized tree



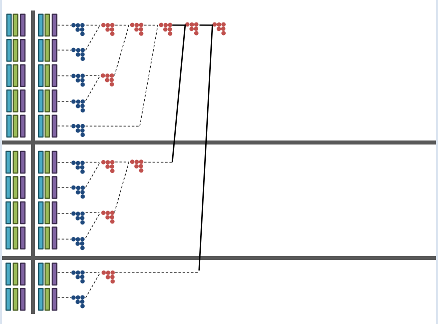
2 inter-cluster communications

Communication pattern (M-by-3 matrix)

ScaLAPACK



TSQR



Articulation of TSQR with QCG-OMPI

ScaLAPACK-based TSQR

- each domain is factorized with a ScaLAPACK call;
- the reduction is performed by pairs of communicators;
- the number of domains per cluster may vary from 1 to 64 (number of cores per cluster).

QCG JobProfile

- processes are split into groups of equivalent computing power;
- good network connectivity inside each group (low latency, high bandwidth);
- lower network connectivity between the groups.

$\longrightarrow$ Classical cluster of clusters approach (with a constraint on the relative size of the clusters to facilitate load balancing).

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3\log_2(C)N^3}$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3\log_2(C)N^3}$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3}\log_2(\mathbf{C})\mathbf{N^3}$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3}\log_2(\mathbf{C})\mathbf{N^3}$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2-2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2-2/3N^3)/C+\mathbf{2/3}\log_2(\mathbf{C})\mathbf{N^3}$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2-4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2-4/3N^3)/C+\mathbf{4/3}\log_2(\mathbf{C})\mathbf{N^3}$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3}\log_2(\mathbf{C})\mathbf{N^3}$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3}\log_2(\mathbf{C})\mathbf{N^3}$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

 Articulation of TSQR with QCG-OMPI

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3}\log_2(\mathbf{C})\mathbf{N^3}$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3}\log_2(\mathbf{C})\mathbf{N^3}$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N}\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3}\log_2(\mathbf{C})\mathbf{N}^3$

Computing Q and R (on C clusters)

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N}\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3}\log_2(\mathbf{C})\mathbf{N}^3$

- C: number of clusters;
- 1 domain per cluster is assumed for these tables.

 Articulation of TSQR with QCG-OMPI

Comm. and computation breakdown (critical path)

Computing R

	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$2\mathbf{N} \log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C$
TSQR	$\log_2(C)$	$\log_2(C)(N^2/2)$	$(2MN^2 - 2/3N^3)/C + \mathbf{2/3 \log_2(C) N^3}$

Computing Q and R (on C clusters)

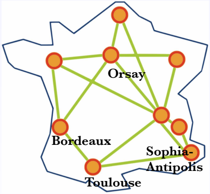
	# inter-cluster msg	inter-cluster vol. exchanged	# FLOPs
ScaLAPACK QR2	$4\mathbf{N} \log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C$
TSQR	$2\log_2(C)$	$2\log_2(C)(N^2/2)$	$(4MN^2 - 4/3N^3)/C + \mathbf{4/3 \log_2(C) N^3}$

- ⋆ C: number of clusters;
- ⋆ 1 domain per cluster is assumed for these tables.

Outline

Experimental environment: Grid'5000

- Four clusters – 32 nodes per cluster – 2 cores per node.
- AMD Opteron 246 (2 GHz/ 1MB L2 cache) up to AMD Opteron 2218 (2.6 GHz / 2MB L2 cache).
- Linux 2.6.30 – ScaLAPACK 1.8.0 – GotoBlas 1.26.
- 256 cores total (theoretical peak 2048 Gflop/s – dgemm upperbound 940 Gflop/s).



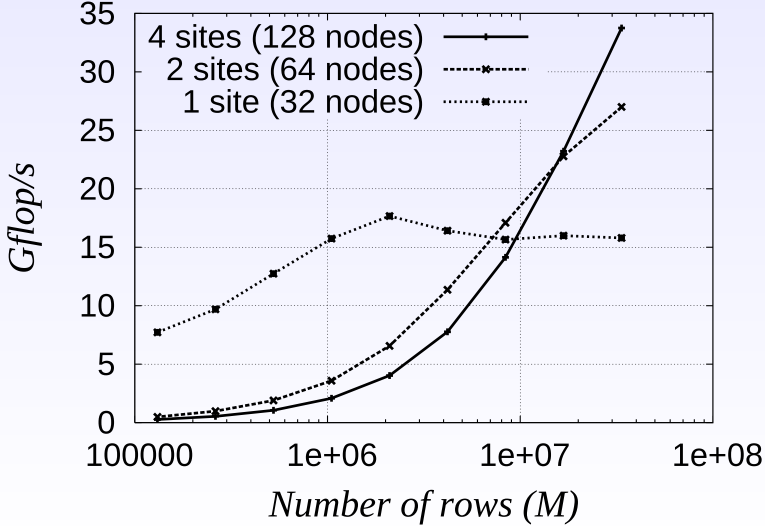
Network

Latency (ms)	Orsay	Toulouse	Bordeaux	Sophia
Orsay	0.07	7.97	6.98	6.12
Toulouse		0.03	9.03	8.18
Bordeaux			0.05	7.18
Sophia				0.06

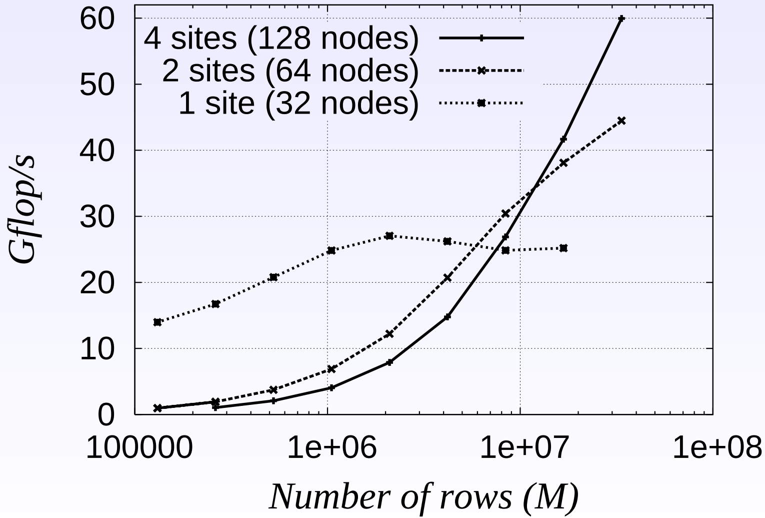
Throughput (Mb/s)	Orsay	Toulouse	Bordeaux	Sophia
Orsay	890	78	90	102
Toulouse		890	77	90
Bordeaux			890	83
Sophia				890

Outline

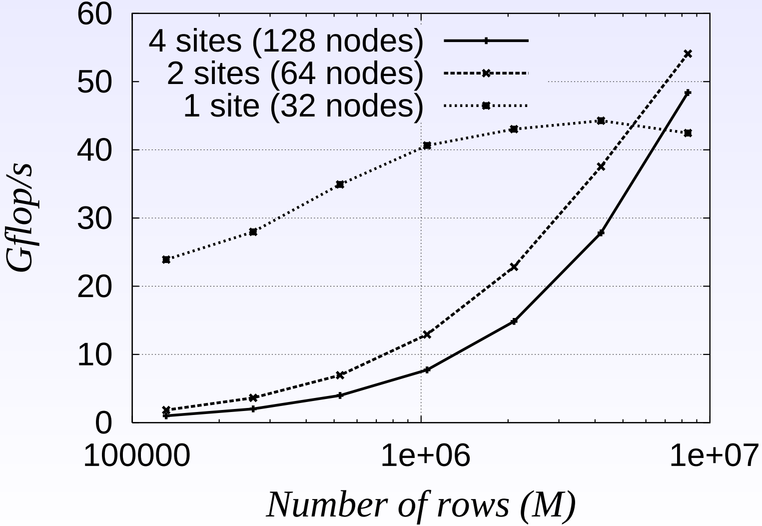
ScaLAPACK - $N = 64$



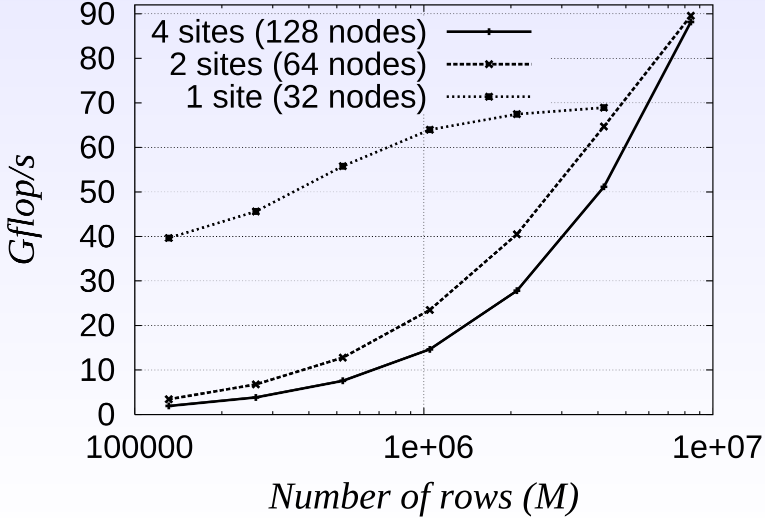
ScaLAPACK - $N = 128$



ScaLAPACK - $N = 256$

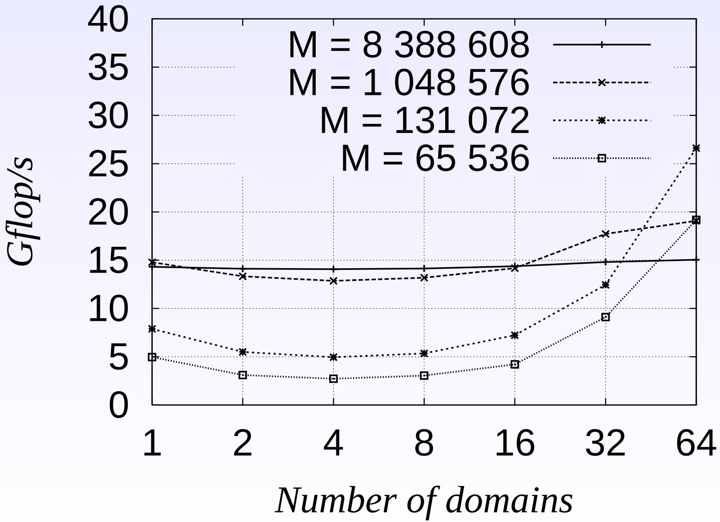


ScaLAPACK - $N = 512$

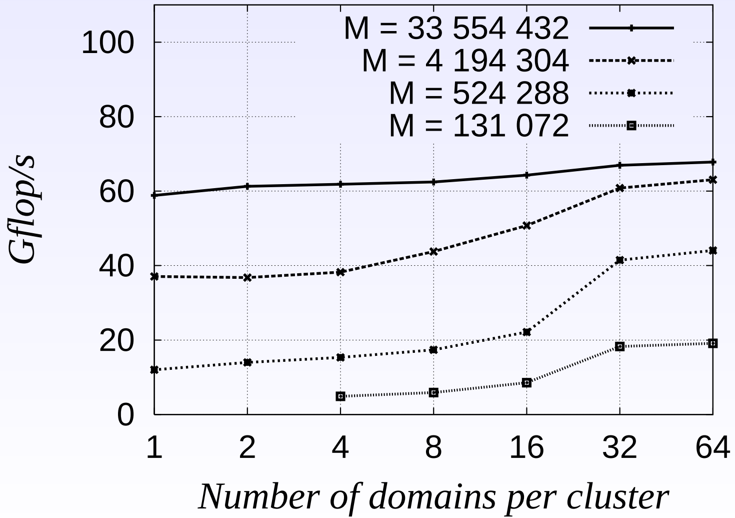


Outline

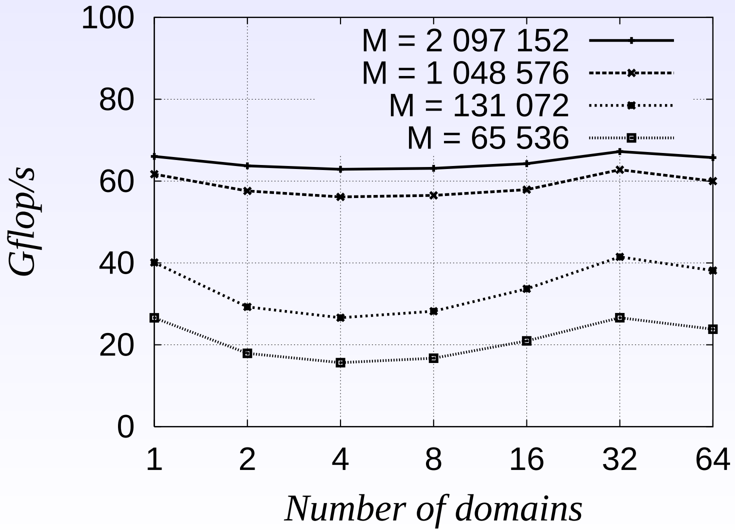
TSQR - $N = 64$ - one cluster



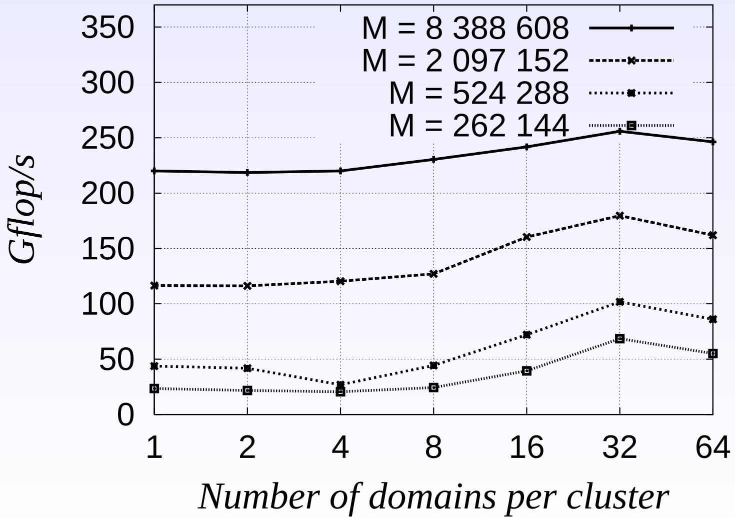
TSQR - $N = 64$ - all four clusters



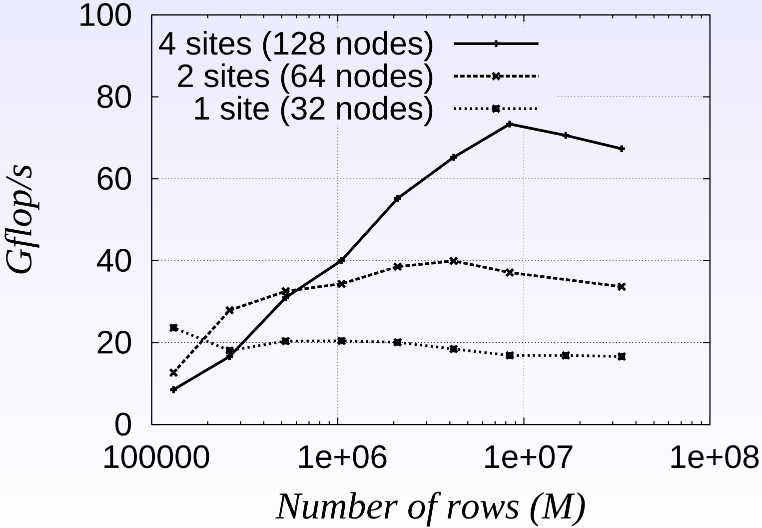
TSQR - $N = 512$ - one cluster



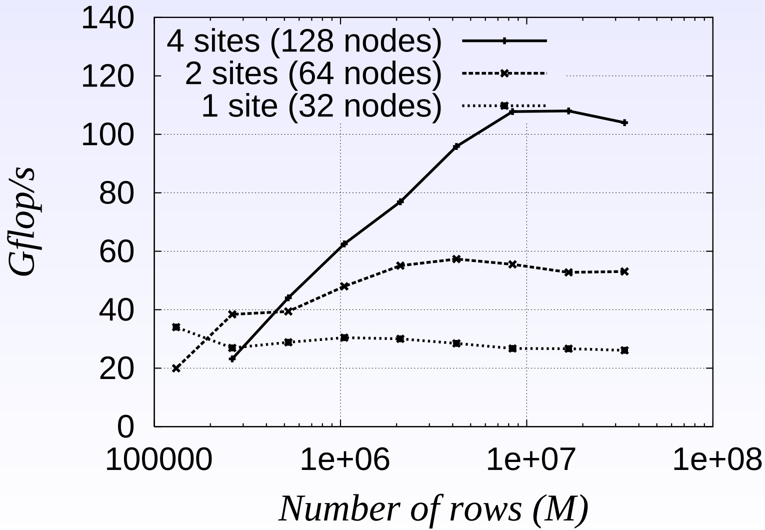
TSQR - $N = 512$ - all four clusters



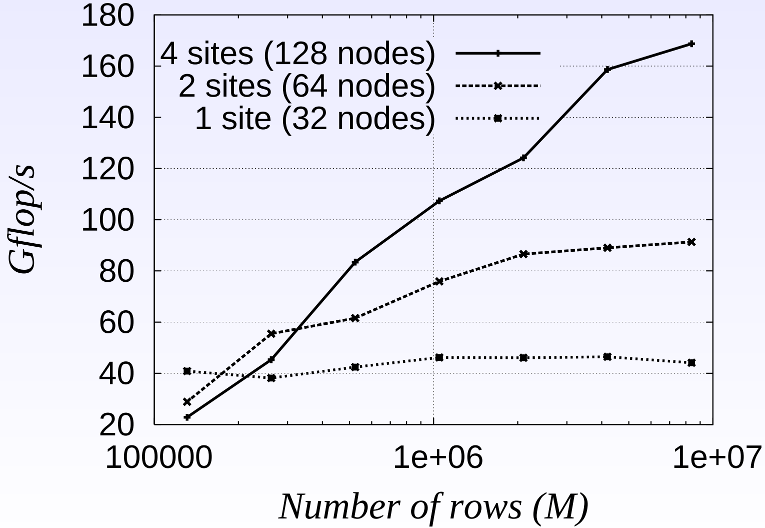
TSQR - $N = 64$


4 sites (128 nodes)
2 sites (64 nodes)
1 site (32 nodes)
Gflop/s
0
20
40
60
80
100
100000
1e+06
1e+07
1e+08
Number of rows (M)

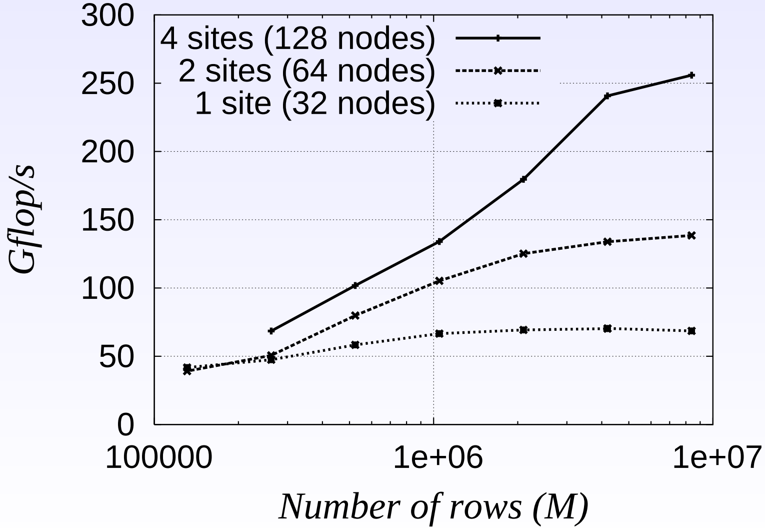
TSQR - $N = 128$


140
120
100
80
60
40
20
0
Gflop/s
4 sites (128 nodes)
2 sites (64 nodes)
1 site (32 nodes)
100000
1e+06
1e+07
1e+08
Number of rows (M)

TSQR - $N = 256$


180
160
140
120
100
80
60
40
20
Gflop/s
4 sites (128 nodes)
2 sites (64 nodes)
1 site (32 nodes)
100000
1e+06
1e+07
Number of rows (M)

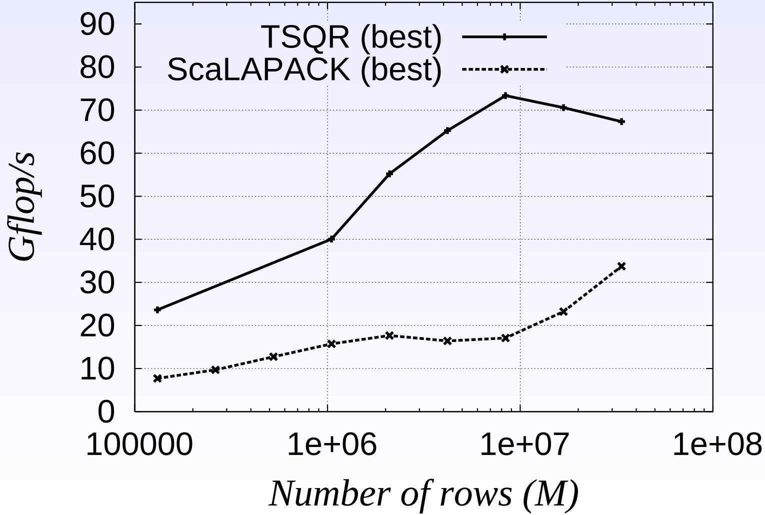
TSQR - $N = 512$



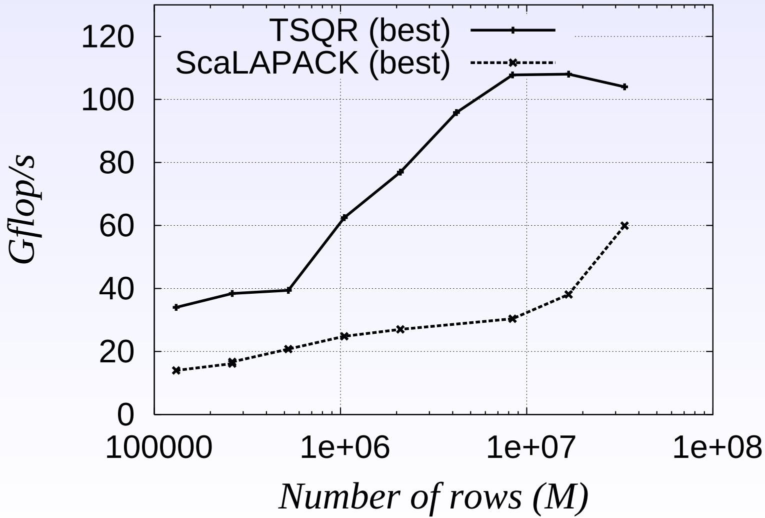
300
250
200
150
100
50
0
Gflop/s
100000
1e+06
1e+07
Number of rows (M)
4 sites (128 nodes)
2 sites (64 nodes)
1 site (32 nodes)

Outline

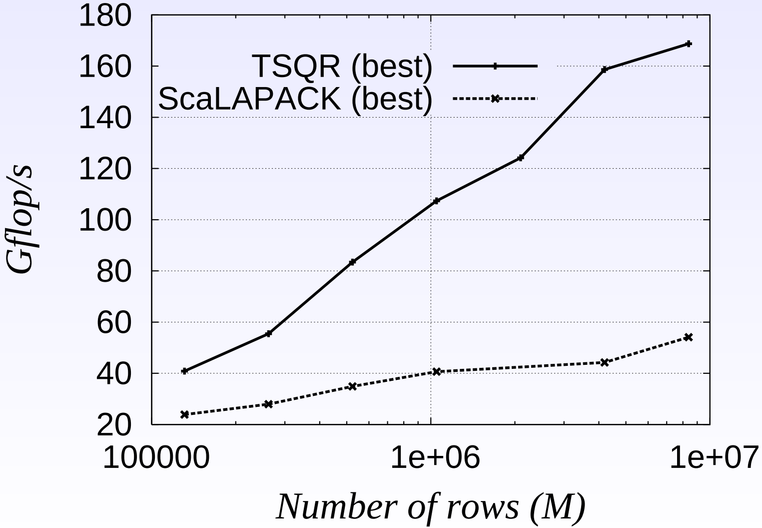
TSQR vs ScaLAPACK - $N = 64$



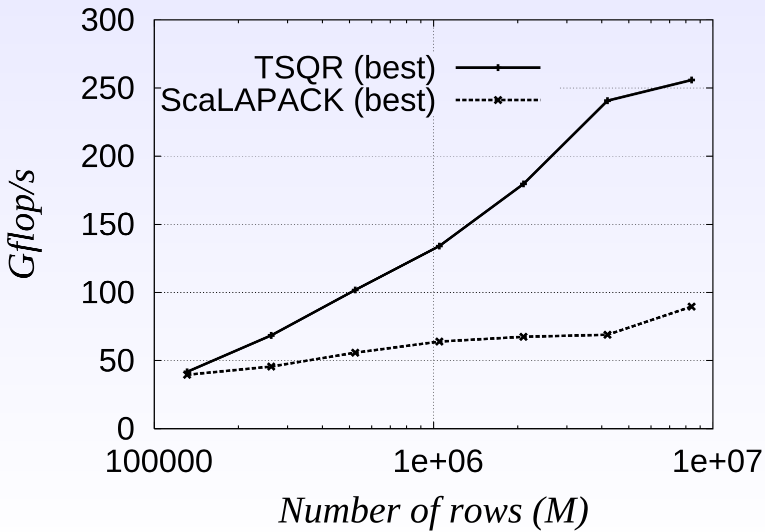
TSQR vs ScaLAPACK - $N = 128$



TSQR vs ScaLAPACK - $N = 256$



TSQR vs ScaLAPACK - $N = 512$



Outline

Conclusion

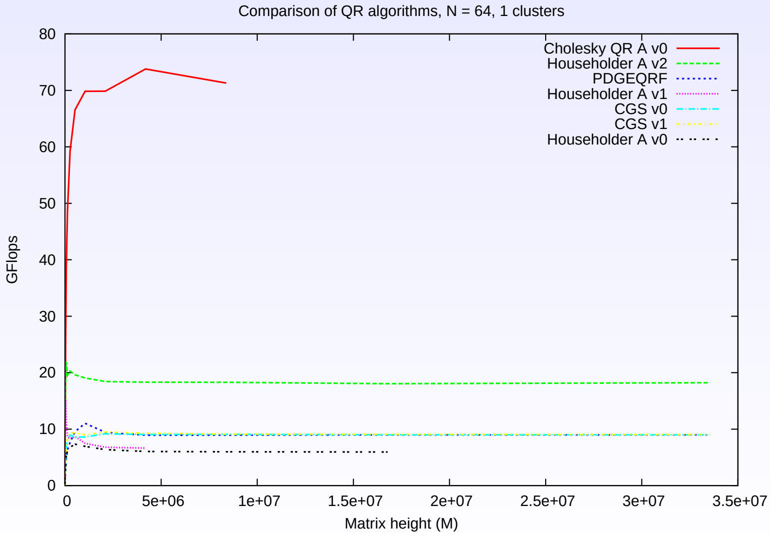
Can we speed up dense linear algebra applications
using a computational grid ?

Yes,
at least for applications based on the QR factorization
of Tall and Skinny matrices.

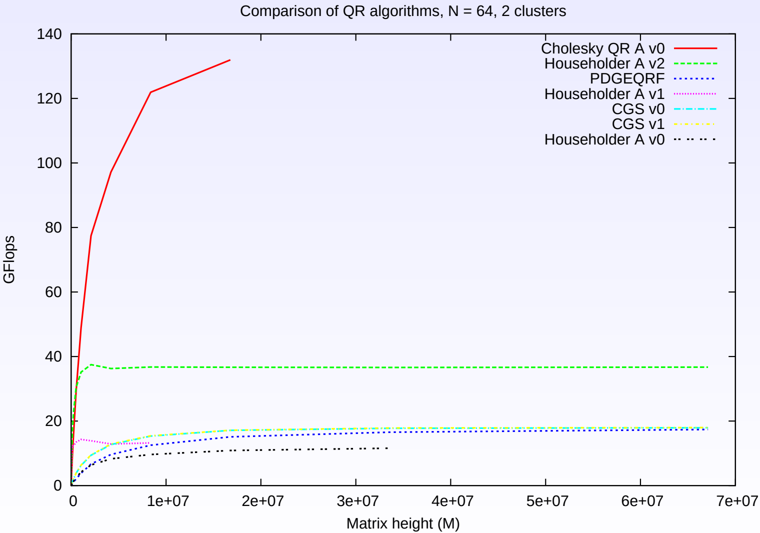
Future directions

- ⋆ What about square matrices (CAQR) ?
- ⋆ LU and Cholesky factorizations ?
- ⋆ Can we benefit from recursive kernels ?

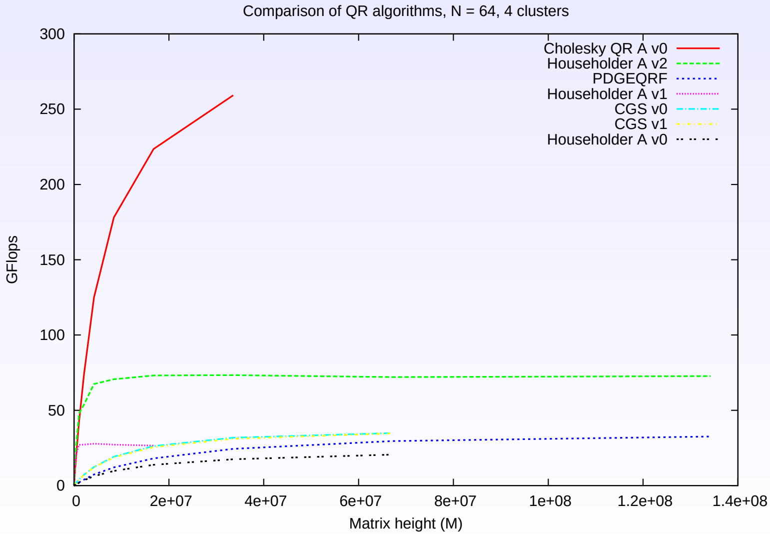
$N = 64$ - one cluster



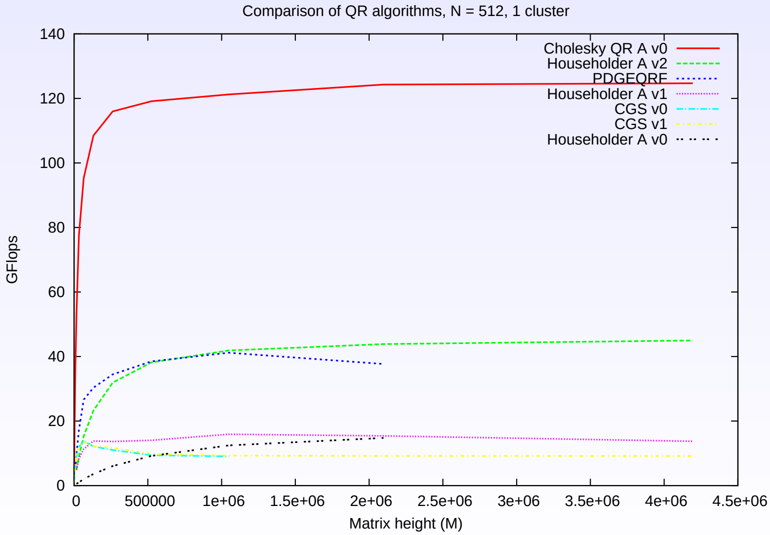
$N = 64$ - two clusters



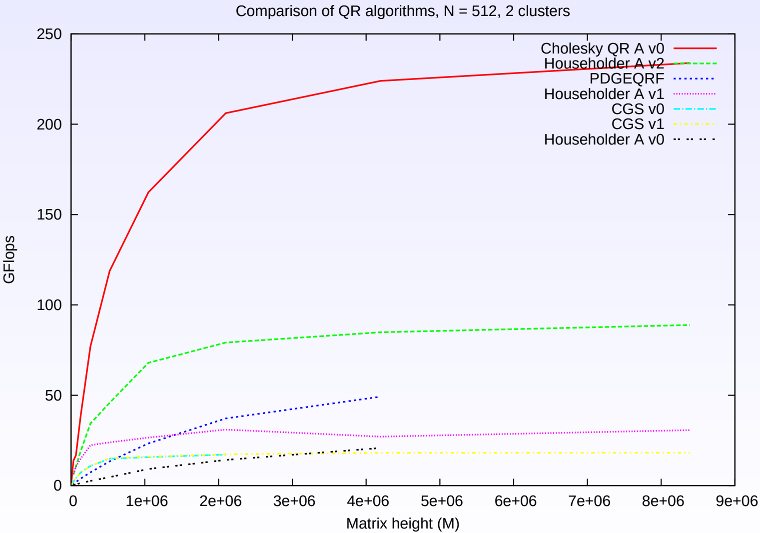
$N = 64$ - all four clusters



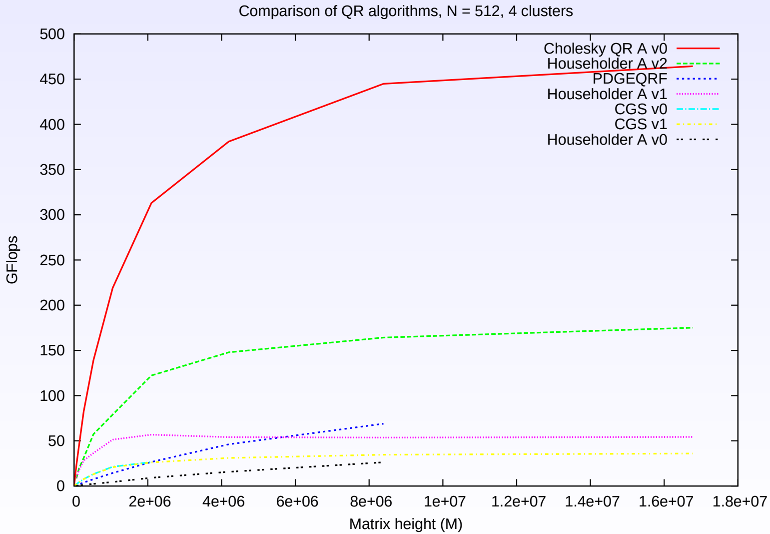
$N = 512$ - one cluster



$N = 512$ - two clusters



$N = 512$ - all four clusters



Thanks

- Questions ?